Available online at www.sciencedirect.com

ScienceDirect

Journal of Electrical Systems and Information Technology 3 (2016) 295–313

www.elsevier.com/locate/jesit

Multi-robot path planning in a dynamic environment using improved gravitational search algorithm

P.K. Das [a,*], H.S. Behera [a], P.K. Jena [b], B.K. Panigrahi [c]

[a] *Dept. of Comp. Sc. and Engineering and Information Technology, VSSUT, Burla, Odisha, India*
[b] *Dept. of Mechanical Engineering, VSSUT, Burla, Odisha, India*
[c] *Dept. of Electrical Engineering, IIT, Delhi, India*

Received 17 August 2015; received in revised form 17 November 2015; accepted 20 December 2015
Available online 2 August 2016

Abstract

This paper proposes a new methodology to optimize trajectory of the path for multi-robots using improved gravitational search algorithm (IGSA) in a dynamic environment. GSA is improved based on memory information, social, cognitive factor of PSO (particle swarm optimization) and then, population for next generation is decided by the greedy strategy. A path planning scheme has been developed using IGSA to optimally obtain the succeeding positions of the robots from the existing position. Finally, the analytical and experimental results of the multi-robot path planning have been compared with those obtained by IGSA, GSA and PSO in a similar environment. The simulation and the Khepera environmental results outperform IGSA as compared to GSA and PSO with respect to performance matrix.

Keywords: Gravitational search algorithm; Multi-robot path planning; Average total trajectory path deviation; Average uncovered trajectory target distance; Average path length

1. Introduction

Gravitational search algorithm (GSA) is effective and efficient using an alternative approach to the multi-robot path planning. Although many algorithms (Tuncer and Yildirim, 2012; Guo and Parker, 2002) have been proposed and proven to be feasible and efficient for robot motion planning and collision avoidance, classic techniques for path planning problem (Konar, 1999; Banerjee et al., 2011) are general methods like Roadmap, Cell Decomposition, Potential Fields, Optical Tweezers and Mathematical Programming. Many authors have proposed multi-robot and the

* Corresponding author.
E-mail addresses: daspradipta78@gmail.com (P.K. Das), hsbehera_india@yahoo.com (H.S. Behera), pjenavssut@gmail.com (P.K. Jena), bkpanigrahi@ee.iitd.ac.in (B.K. Panigrahi).
Peer review under the responsibility of Electronics Research Institute (ERI).

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jesit.2015.12.003

single robot path planning problems using different classical techniques (Kcymeulcn and Decuyper, 1994; Li et al., 2009), Neural Network (Yu and Kromov, 2001), artificial immune system (Das et al., 2012; Luh and Cheng, 2002) and heuristic optimization algorithms (Das et al., 2010, 2011; Geem et al., 2001; Yang, 2009; Regele and Levi, 2006). High time complexity in large problem spaces and trapping in local optimum are drawbacks for classic techniques and in many meta-heuristic algorithms. These drawbacks cause the classical techniques and inefficient in the various problem spaces. To improve the efficiency of classical methods, probabilistic algorithms like PRM and RRT are proposed for improving the local optimization problem, many evolutionary algorithms like Genetic Algorithms (Tuncer and Yildirim, 2012; Gong and Lincheng, 2001), PSO (Zhang et al., 2013; Masehian and Sedighizadeh, 2010), bee colony optimization (Bhattacharjee et al., 2011) and differential evolution algorithm (Chakraborty et al., 2009) are used in multi-robot path planning problem.

The gravitational search algorithm (Verma et al., 2013; Eldos and Qasim, 2013; Chatterjee et al., 2011) is a recent algorithm that has been inspired by the Newtonian's law of gravity and motion. GSA has undergone a lot of changes to the algorithm itself and has been applied in various applications. At present, there are various variants of GSA (Precup et al., 2012; Rashedi et al., 2010, 2009b; Purcaru et al., 2013) which have been developed to enhance and improve the original version. The algorithm has also been explored in many areas (Sabri et al., 2013; Eldos and Qasim, 2013).

For realization multi robot path planning problem with different goal of the respective robots with GSA (Precup et al., 2012; Tuncer and Yildirim, 2012) by the centralized approach, a fitness function is constructed to determine the next position of the robots that lie on optimal trajectories leading toward the respective goals. The fitness function of the GSA (Alba and Dorronsoro, 2005) has two main components: first one is the objective function describing the selection of next position on an optimal trajectory based on velocity, and the second one is the constraint on acceleration representing avoidance of collision with other robots and with static obstacles. The path-planning problem considered here is formulated by a centralized approach, where an iterative algorithm is invoked to determine the next position of all the robots satisfying all the constraints imposed on the multi-objective function. The algorithm is iterated until all the robots reach their destination (goal position).

The advantages of GSA are (1) easy to implement with higher computational efficiency; (2) few parameters to adjust, but the disadvantages of this algorithm are as follow (1) if premature convergence occurs, there will not be any recovery for this algorithm; (2) the algorithm loses its ability to explore and then becomes inactive only after becomes convergence. Due to above difficulties in GSA, further improvements are required for the optimal solution to the complex problem. Here, we consider the improvement of GSA which is based on the communication and memory characteristics of PSO (particle swarm optimization). Therefore, we called it improved gravitational search algorithm.

The main objective of this paper is summarized as follows: (i) we study the problem of multi-robot path planning in a clutter environment and formulated the above problem as multi-objective optimization problem with constraints; (ii) a novel method to the solution of an optimal trajectory path generation for multi robot path planning problem using IGSA is proposed in this article; (iii) the proposed algorithm has been applied for multi robot path planning in a clutter and dynamic environment and obtained results are compared to other optimization algorithms like GSA, DE; (iv) the performance of the proposed IGAS, as an optimizing tool in solving multi robot path planning problem, is applied in the simulation as well as Khepera-II environment and result is presented; (v) the performance matrix of the proposed approach is successfully validated in simulation and Khepera-II.

In this paper, the implementation of the modified gravitational search technique has been proposed to determine the trajectory path for multiple robots from predefined initial positions to predefine target positions in the environment with an objective to minimize the path length of all the robots. The result shows that the algorithm can improve the solution quality in a reasonable amount of time and also improves the convergence rate. This paper improves the gravitational search algorithm (IGSA) for improving the global path planning problem of the multi-robots by improving the convergence rate. Finally, the efficiency of the IGSA has been proved through the simulation as well as Khepera environment and a result obtained is compared with other evolutionary computing such an GSA and DE.

The rest of the paper is outlined as follows: Section 3 briefly describes the improved gravitational search algorithm. Formulation of the problem for multi-robot path planning has been elaborated in Section 4. Multi-objective optimization problem solving using improved GSA is described in details in Section 5. Section 6 demonstrates the result of path planning for multi-robot through simulation. In Section 7, the experiment has been conducted in Khepera II environment and finally, the conclusion of the work is presented in Section 8.

2. Gravitational search algorithm (GSA)

Recently, the scientific community has gained the interest on GSA. It is a meta-heuristic optimization algorithm inspired by nature which is based on the Newton's law of gravity and the law of motion (Rashedi et al., 2009a; Sabri et al., 2013). GSA is grouped under the population based approach and is reported to be more natural. The algorithm has been planned to improve the performance in the exploration and manipulation capabilities of a population based algorithm, based on gravity rules.

GSA is based on the two important formulas about Newton gravity laws given by Eqs. (1) and (2). Eq. (1) is the gravitational force equation between the two particles, which is directly proportional to their masses and inversely proportional to the square of the distance between them. But in GSA instead of the square of the distance, only the distance is used. Eq. (2) is the equation of acceleration of a particle when a force is applied to it (Rashedi et al., 2009a; Sabri et al., 2013).

$$F = G\frac{M_1 M_2}{R^2} \tag{1}$$

$$a = \frac{F}{M} \tag{2}$$

G is gravitational constant, M_1 and M_2 are masses and R is the distance between them, F is gravitational force, and a is acceleration. Based on these formulas, the heavier object with more gravity force attracts the other objects as it is seen in Fig. 1.

In GSA, each mass (agent) has four characteristics, namely: position, inertial mass, active gravitational mass, and passive gravitational mass. The position of the mass corresponds to a solution of the problem, and the fitness function is used to determine the gravitational and inertial masses (Verma et al., 2013; Sabri et al., 2013). The more precisely masses obey the following laws.

Law of gravity: Each particle attracts every other particle and the gravitational force between two particles is directly proportional to the product of their masses and inversely proportional to the distance between them, R. We use here R instead of R^2, because according to our experimental results, R provides better results than R^2 in all experimental cases.

Law of motion: The current velocity of any mass is equal to the sum of the fraction of its previous velocity and the variation in the velocity. Variation in the velocity or acceleration of any mass is equal to the force acted on the system divided by mass of inertia.

2.1. Agents initialization

Consider a system with N masses in which position of the i^{th} mass is defined as follows:

$$X_i = \left(x_i^1, \ldots x_i^d, \ldots, x_i^n\right) \quad \text{for } i = 1, 2, \ldots N \tag{3}$$

where x_i^d is the position of i^{th} mass in d^{th} dimension and n is the dimension of the search space.

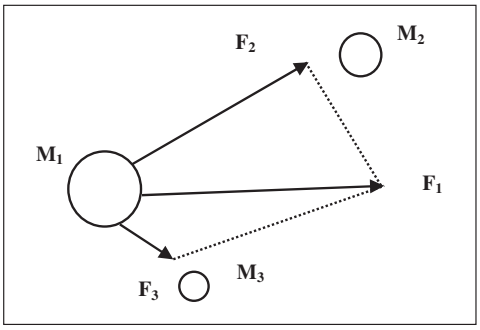

Fig. 1. The Newton gravitational force representation.

2.2. Fitness and best fitness computation

worst(t) and best(t) are defined as follows for the minimization case:

$$\text{worst}(t) = \max_{i \in p} \text{fit}_i(t), \quad p = 1, 2, \ldots, N \tag{4}$$

$$\text{best}(t) = \min_{i \in p} \text{fit}_i(t), \quad p = 1, 2, \ldots, N \tag{5}$$

2.3. Gravitational constant (G) computation

The gravitational constant G is computed at iteration (Sabri et al., 2013).

$$G(t) = G_o e^{(-\alpha t/T)} \tag{6}$$

Here, T is the maximum iteration, t is the current iteration and $\alpha \succ 0$ is the weight factor, computed as follows.

$$\alpha = \alpha_{\max} - \frac{\alpha_{\max} - \alpha_{\min}}{T} \times t \tag{7}$$

2.4. Masses of the agents' calculation

Each agent's mass is calculated after computing current population's fitness as:

$$m_i(t) = \frac{\text{fit}_i(t) - \text{worst}(t)}{\text{best}(t) - \text{worst}(t)} \tag{8}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \tag{9}$$

where $M_i(t)$ and $\text{fit}_i(t)$ represent the mass and the fitness value of the agent i at iteration t, respectively.

2.5. Velocity and positions of agents

The velocity and position of the agents are updated as:

$$V_i^d(t+1) = \beta V_i^d(t) + a_i^d(t) \tag{10}$$

$$x_i^d(t+1) = x_i^d(t) + V_i^d(t+1) \tag{11}$$

Here, β is the random number, $0 \leq \beta \leq 1$ and an acceleration of the i^{th} agents at iteration 't' is computed as,

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)} \tag{12}$$

$F_i^d(t)$ is the total force acting on i^{th} agent calculated as:

$$F_i^d(t) = \sum_{j \in k_{\text{best}}, j \neq i} \beta F_{ij}^d(t) \tag{13}$$

K_{best} is the set of first K agents with the best fitness value and biggest mass, which is a function of time, initialized to k_0 at the beginning and decreasing with time. Here k_0 is set to N (total number of agents) and is decreased linearly to 1.

$F_{ij}^d(t)$ is computed using the following equation:

$$F_{ij}^d(t) = G(t) \times \left(M_{pi}(t) \times \frac{M_{aj}(t)}{\text{dis}_{ij}(t) + \varepsilon} \right) \times \left(X_j^d(t) - X_i^d(t) \right) \tag{14}$$

Here X_i and X_j are the position vector of the i^{th} and j^{th} agent in d^{th} dimension, $F_{ij}^d(t)$ is the force acting on agent i from agent j at d^{th} dimension and i^{th} iteration. $\text{dis}_{ij}(t)$ is the Euclidian distance between two agents i and j at iteration t. $G(t)$ is the computed gravitational constant at the same iteration while ε is a small constant. $M_{pi}(t)$ is the passive

gravitational mass of the agent i at the instance t. $M_{aj}(t)$ is the active gravitational mass of the agent j at time t, these masses being calculated according to Precup et al. (2012), Rashedi et al. (2010, 2009b) and Purcaru et al. (2013).

3. Improvement of the gravitational search algorithm based on PSO and greedy strategy

Most of the meta-heuristic searching algorithm find its best solution due to good balance of exploration and exploitation (Alba and Dorronsoro, 2005; Liu et al., 2013). The exploration capability of the algorithm provides the connectivity relationship of the search space, which helps to find global optimal solution. The exploitation helps to find the better optimal solutions in the visited domain, which reinforce the convergence capability of local search. So, good meta-heuristic algorithm should improve the exploration ability in the first step and then exploitation ability with increasing of iteration in second step. Therefore, the gravitational search algorithm has been improved to maintain the good balance between exploration and exploitation. In GSA, the moment direction of each agent is based on the total force act by other agents on it and lacking the communication between the agents. Therefore, improvement of the searching ability of GSA based on the memory and social information of PSO and to accelerate the convergence rate, weight value is assigned to inertia mass of every agent in each iteration (Sarafrazi et al., 2011) and then, optimized solution saving strategy is used with reference to DE (Sarafrazi et al., 2011). The PSO updates the velocity using the cognitive and social factor. The velocity and position update equation of PSO are as follow:

$$V_i^d(t+1)_{\mathrm{PSO}} = wV_i^d + C_1 \times \varphi_1 \times (x^d_{p\mathrm{best}_i} - x_i^d(t)) + C_2 \times \varphi_2 \times (x^d_{g\mathrm{best}} - x_i^d(t)) \tag{15}$$

$$x_i^d(t+1)_{\mathrm{PSO}} = x_i^d(t+1) + V_i^d(t+1) \tag{16}$$

$$v_i^d(t+1)_{\mathrm{GSA}} = \beta v_i(t) + a_i^d(t) \tag{17}$$

where Eq. (17) is the GSA velocity formulation obtained from Eq. (10). In this paper, GSA is improved by adopting the memory, social and cognitive information of PSO. The velocity updating equation in GSA can be defined as

$$V_i^d(t+1)_{\mathrm{IGSA}} = \beta V_i^d(t)_{\mathrm{GSA}} + a_i^d(t) + C_1 \times \varphi_1 \times (x^d_{p\mathrm{best}_i} - x_i^d(t)) + C_2 \times \varphi_2 \times (x^d_{g\mathrm{best}} - x_i^d(t)) \tag{18}$$

$$x_i^d(t+1)_{\mathrm{IGSA}} = x_i^d(t) + V_i^d(t+1)_{\mathrm{IGSA}} \tag{19}$$

where Eq. (19) is the IGSA velocity formulation, which is formulated and updated using PSO velocity by considering the memory, social and cognitive factor and GSA acceleration. C_1 and C_2 balance the effectiveness of "law of gravity and memory and social information". The optimized solution saving strategy is used for deciding the member for next generation $t+1$ with reference to differential evolution (DE) (Sarafrazi et al., 2011). The "survival of fittest" strategy is used to decide the member for next generation. Here, the greedy strategy has been devised for deciding better target vector. The population for next generation is decided by comparing the trial vector $x_i^d(t+1)$ with the target vector $x_i^d(t)$. The selection procedure can be expressed by the following expression.

$$x_i^d(t+1) = \begin{cases} x_i^d(t), & \text{if } \mathrm{fit}(x_i^d(t)) < \mathrm{fit}(x_i^d(t+1)) \\ x_i^d(t+1), & \text{otherwise} \end{cases} \tag{20}$$

4. Problem formulation for multi robot path planning

The problem formulation for multi-robot path planning is to determine the next position of the robot from their existing positions in its workspace by avoiding the collision with other robots and obstacles (which are static in nature) in its path to reach at the goal. Multi-robot path planning problem is formulated by considering the set of principles using the following assumptions by a uniform treatment.

Assumptions

a For each robot the current position (recent position) and goal position (target position) is known in a given reference coordinate system.
b The robot can select any action in a given time from a fixed set of actions for its motions.
c Each robot is performing its action in steps until all robots reached in their respective target positions.

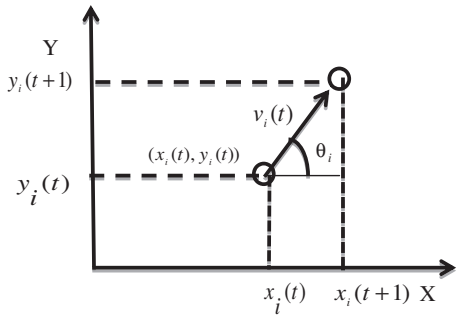

Fig. 2. Representation of next position from current position of the *i*-th robot.

The following principles have been taken care for satisfying the given assumptions.

1. For determining the next position from its current position, the robot tries to align its heading direction toward the goal.
2. The alignment may cause a collision with the robots/obstacles (which are static in nature) in the environment, hence, the robot has to turn its heading direction left or right by a prescribed angle to determine its next position.
3. If a robot can align itself with a goal without collision, then, it will move to that determine the position.
4. If rotating the heading direction left or right requires the same angle of rotation of the robot about the *z*-axis, if it is tied then, broken randomly.

Consider the initial position of the *i*th robot at a time *t* is $(x_i(t), y_i(t))$, the next position of the same robot at a time $(t + \delta t)$ is $(x_i(t + \delta t), y_i(t + \delta t))$, $v_i(t)$ is the velocity of the robot R_i and $(x_i^{\text{goal}}, y_i^{\text{goal}})$ is the target or goal position of the robot R_i.

So, the expression for the next position $(x_i(t + \delta t), y_i(t + \delta t))$ can be derived from Fig. 2 as follows

$$x_i(t + \delta t) = x_i(t) + v_i(t)\cos\,\theta_i \delta t \tag{21}$$

$$y_i(t + \delta t) = y_i(t) + v_i(t)\sin\,\theta_i \delta t \tag{22}$$

When $\delta t = 1$, Eqs. (21) and (22) are reduced to

$$x_i(t + 1) = x_i(t) + v_i(t)\cos\,\theta_i \tag{23}$$

$$y_i(t + 1) = y_i(t) + v_i(t)\sin\,\theta_i \tag{24}$$

Consider initially, the robot R_i is placed in the location at $(x_i(t), y_i(t))$. We want to find the next position of the robot $(x_i(t + 1), y_i(t + 1))$, such that the line joining between $\{(x_i(t), y_i(t)), (x_i(t + 1), y_i(t + 1))\}$ and $\{(x_i(t + 1), y_i(t + 1)), (x_i^{\text{goal}}, y_i^{\text{goal}})\}$ should not touch the obstacle in the world map is represented in Fig. 3 and minimizes the total path length from current position to a goal position without touching the obstacle by forming constraint. Then

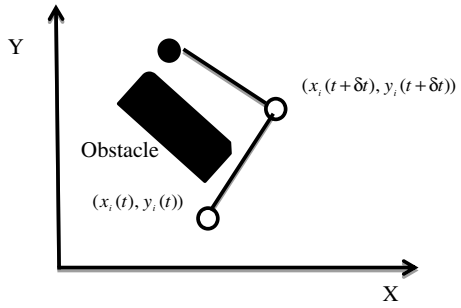

Fig. 3. Selection of next position $(x_i(t + \delta t), y_i(t + \delta t))$ from the current position $(x_i(t), y_i(t))$ to avoid collision with obstacles.

objective function fit_1 that determines the length of the trajectory for n number of robots,

$$\text{fit}_1 = \sum_{i=1}^{n} \left\{ \sqrt{((x_i(t) - x_i(t+1))^2 + (y_i(t) - y_i(t+1)^2)} + \sqrt{((x_i(t+1) - x_i^{\text{goal}})^2 + (y_i(t+1) - y_i^{\text{goal}})^2)} \right\} \tag{25}$$

By putting the value ${x_i}^{\text{next}}$ and y_i^{next} from Eqs. (21) and (24) into Eq. (25), we obtain,

$$\text{fit}_1 = \sum_{i=1}^{n} \left\{ v_i(t) + \sqrt{(x_i(t) + v_i(t)\cos\,\theta_i - x_i^{\text{goal}})^2 + (y_i(t) + v_i(t)\sin\,\theta_i - y_i^{\text{goal}})^2} \right\} \tag{26}$$

The multi-robot path-planning is now represented as an optimization problem by minimizing the objective function in Eq. (26) with considering the penalty function as the constraints in the objective function. Minimizing the objective function in Eq. (26) shows that the robot will follow the shortest distance from the initial point to target point. The constraints here are two types of penalty. The first penalty is to avoid collision between teammates (any two mobile robots), whereas the second penalty is to avoid collision of a robot with a static obstacle. By combining these two penalties a linear fuzzy function is developed for evaluating the obstacle present in the path. So, the objective function formed based on the fuzzy function is denoted as fit_j.

$$\text{fit}_j = \begin{cases} 1 & d(O_j) \le d(O_j)_{\min} \\ \exp\left(-\alpha \dfrac{d(O_j) - d(O_j)_{\min}}{d(O_j)_{\max} - d(O_j)_{\min}}\right) & d(O_j)_{\min} \prec d(O_j) \prec d(O_j)_{\max} \\ 0 & \text{d}(O_j) \ge d(O_j)_{\max} \end{cases} \tag{27}$$

where α is a positive constant, $d(O_j)$ be the distance between mobile robot and obstacles, $d(O_j)_{\max}$ is maximum distance and $d(O_j)_{\min}$ is the minimum distance with respect to the obstacle O_j. The path is safe and collision free path, when $d(O_j) \ge d(O_j)_{\max}$ and path is unsafe if, $d(O_j) \ge d(O_j)_{\min}$.

Thus, mathematically, the optimization problem for obstacles can be formulated as follows:

$$\text{fit}_2 = \max_{j=1,2,\ldots N}(\text{fit}_j) \tag{28}$$

Thus, the optimization problem can be represented as follows:

$$\text{fit} = \text{fit}_1 + \frac{\lambda}{\text{fit}_2} \tag{29}$$

Here, λ is positive constant. The above optimization problem is to minimize the Euclidean distance between the current position and their goal position which is presented by the objective function fit_1 and the second objective function is a constraint to find the safe path.

5. Multi-objective optimization problem solving using IGSA

In this section, multi-robot path planning algorithm has been proposed using IGSA. The proposed IGSA algorithm is used to evaluate the next positions of every robot by presuming the current positions of robots and speeds as the parameter for optimizing the given multi-objective function. It determines the optimized path from each state to the goal state in both dynamic and static environments and the robot measures its distance to obstacles with the help of equipping sensors.

The agents are moving in the search space with the help of the gravity is considered in the proposed IGSA based path planning. The outline of the proposed algorithm is discussed below:

Procedure IGSA (x_{curr_i}, y_{curr_i}, pos-vector)

Begin

Initialize positions of N agents randomly in the population and gravitational constant G_o, C_1,w, C_2, α_{max}, α_{min}, λ, Maxiter, β, N

Set $K_{best} = N$

For d= 1 to no_of _dimension

For iter = 1 to Maxiter do

Begin

Repeat

For each agent

Begin

a. Evaluate the fitness of each agent using (29)

b. The total force on an agent, i by other agents is $F_i^d(t)$ is calculated as:

$F_i^d(t) = \sum_{j=1, j\neq i}^{N} \beta F_{ij}(t)$ and $F_{ij}(t)$ is calculated using (14)

c. Calculate $worst(t)$ and $best(t)$ using (4) and (5)

d. Compute $M_i(t)$, G(t) of agent by (9) and (6)

e. Update the α by (7)

f. Compute the acceleration of each agent using (12)

g. Compute the velocity of each agent using (18)

h. Update the position of agent by (19)

i. Execute the selection procedure for next generation by(20)

End for

Until $K_{best} \neq 1$

End for

End for

Update :

$$x_{curr-i} = x_{curr-i} + V_i^d \cos\theta_i$$

$$y_{curr-i} = y_{curr-i} + V_i^d \sin\theta_i$$

Return

End

Pseudo Code for path planning

Input: $(x_i(t), y_i(t))$,(x_i^{goal}, y_i^{goal}) and $v_i(t)$ are the initial position, goal position and velocity for n robots, $1 \le i \le n$ respectively and ε is the threshold value.

Output: Trajectory of path TP_i for each robot R_i from $(x_i(t), y_i(t))$ to (x_i^{goal}, y_i^{goal})

Begin

For *i = 1 to n*

$x_{curr\_i} \leftarrow x_i(t)$; $y_{curr\_i} \leftarrow y_i(t)$;

End for

Repeat

For robot $i = 1$ to n

Call IGSA (x_{curr_i}, y_{curr_i}, pvector);

// pvector is the position vector denotes updated current position of the i-th robot //

End for;

For $i = 1$ to n

Move-to (x_{curr_i}, y_{curr_i});

End for;

Until $\|curr\_i - G_i\| \le \varepsilon$

// $_{curr\_i}$ = (x_{curr_i}, y_{curr_i}), $G_i = (x_i^{goal}, y_i^{goal})$//

End.

6. Computer simulation

The multi-robot path-planning algorithm is implemented in a simulated environment. The simulation is conducted in a C environment on a Pentium processor and the experiment was performed with 14 robots of circular shape. The radius of each robot is considered as 6 pixels. Before initiating the experiment for multi-robot path planning, each robot starting and goal points are predefined. The experiments were performed with seven differently shaped obstacles and with equal velocities for all the robots in a given run of the program; however, the velocities were adjusted over different runs of the same program. One of our experimental world-maps with an initial configuration of the world-map with 7 obstacles and the starting and the goal positions of 12 circular soft-bots are shown in Fig. 4. The intermediate steps of movement of the robots are shown in Figs. 5 and 6. The final stage of world maps, where all the robots reached in their predefine goal respectively is shown in Fig. 7. The simulation is also conducted in the environment as presented in Fig. 4 for same number of robots by GSA and DE. The optimal trajectory of the path has been presented in Figs. 8 and 9 for GSA and DE respectively.

The experiment has been conducted using a central version of the algorithm using the fitness Function (29) for deciding the next position of the robot. In our experiment, parameters have been described in Table 1 for simulation and Khepera II environment.

6.1. Average total trajectory path distance (ATTPD)

Consider a trajectory path from the predefine starting point S_k to the goal point G_k generated by the program for robot R_k in the jth run is TP_{kj}. If $TP_{k1}, TP_{k2}, \ldots, TP_{kj}$ are the trajectory paths generated over j^{th} runs, then the average total trajectory path traversed (ATTPT) by a robot R_k is given by $\sum_{r=1}^{j} TP_{ir}/j$ and the average trajectory path distance

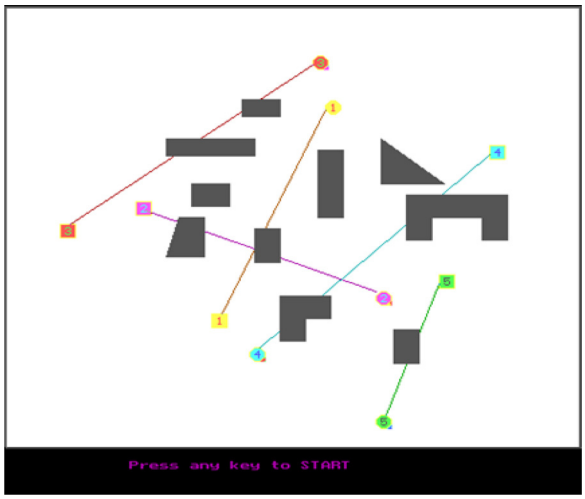

Fig. 4. Initial world map with 7 obstacles and 5 robots.

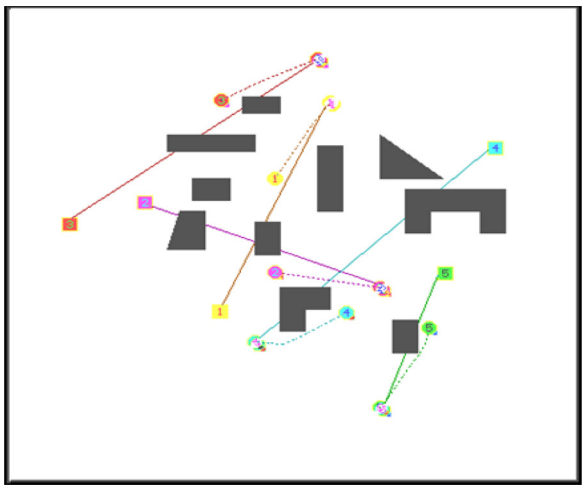

Fig. 5. Intermediate state of the world map during execution using IGSA for 5 robots and 7 obstacles after 9 steps.

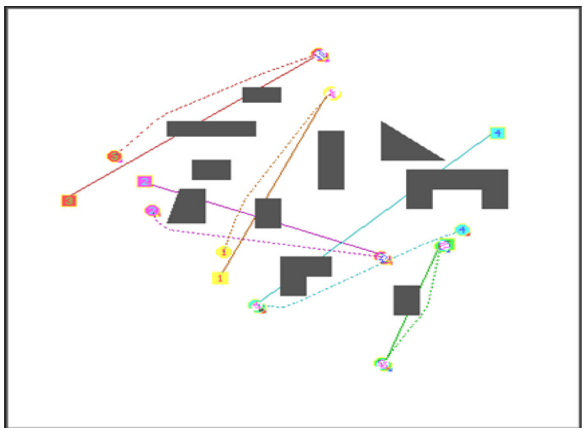

Fig. 6. Intermediate stage of the world map during execution of IGSA for 5 robots and 7 obstacles after 17 steps.

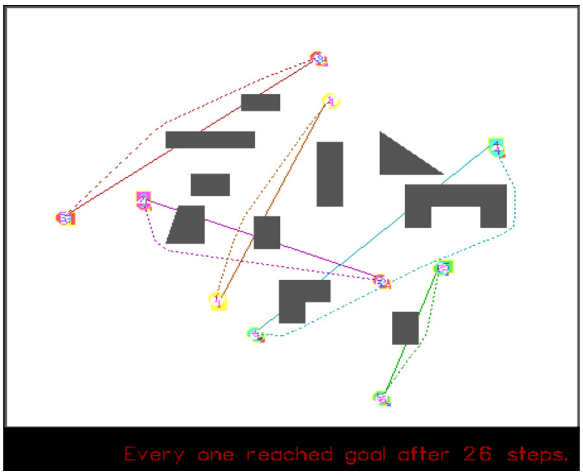

Fig. 7. Five robots reached in their respective pre-defined goal.

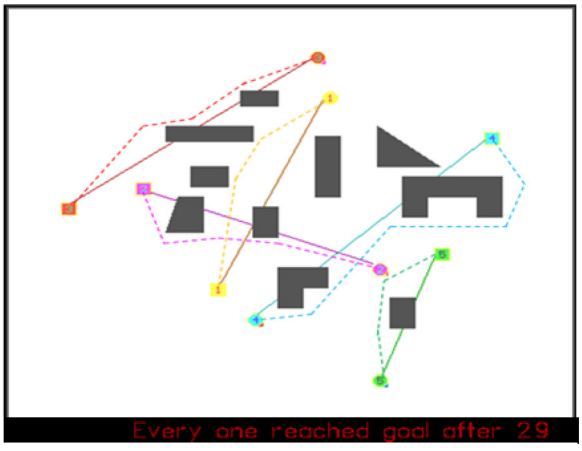

Fig. 8. All robots reached in their respective pre-defined goal in 29 steps by GSA.

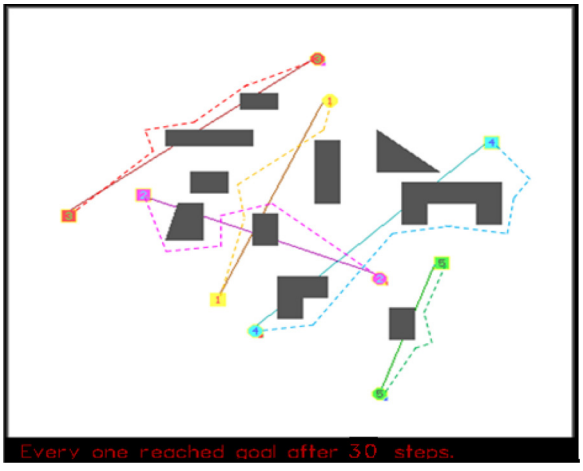

Fig. 9. All robots reached in their respective pre-defined goal in 30 steps by PSO.

Table 1
Parameter used in the simulation and Khepera.

Parameters	Values
G_0	100
$\alpha_{\min}$	0.2
$\alpha_{\max}$	0.4
λ	100
C_1	0.5
C_2	0.5
T (Maxiter)	100
W	0.72
β	0.5
N	50

for this robot is evaluated by measuring the difference between ATPT and the ideal shortest path between S_k to G_k. If the ideal trajectory path for robot R_k obtained geometrically is $\mathrm{TP}_{k-\mathrm{real}}$, then the average path distance is given by

$$\mathrm{TP}_{k-\mathrm{real}} - \sum_{r=1}^{j} \frac{P_{ir}}{j}.$$

Therefore, for n robots in the workspace the average total path distance (ATPD) is

$$\sum_{i=1}^{n} \left(\mathrm{TP}_{k-\mathrm{real}} - \sum_{r=1}^{j} \frac{P_{ir}}{j} \right)$$

6.2. Average uncovered trajectory target distance (AUTTD)

Given a goal position G_k and the current position C_k of a robot on a 2-dimensional workspace, where G_k of C_k are 2-dimensional vectors, the uncovered trajectory distance for the robot k is $\|G_k - C_k\|$, where $\|.\|$ denotes Euclidean norm. For n robots, uncovered trajectory target distance (UTTD) is $\mathrm{UTTD} = \sum_{i=1}^{n} \|G_k - C_k\|$. For k runs of the program, we evaluate the average of UTTDs, and call it the average uncovered trajectory target distance (AUTTD). Fig. 16 shows that by decreasing the velocity, AUTTD takes longer time to converge and gradually terminated with iteration. Again, it is noted that larger the velocity of the robot, the faster falloff in the AUTTD. Fig. 17 shows that, larger the number of robots, slower the convergence rate. Slower the convergence causes the delay in falloff in AUTTD.

The performance analysis was undertaken in the simulation environment and the ATPT was ploted for n robots, called average total trajectory path traveled (ATTPT) by varying no. of robots 1–5 presented in Fig. 18 and generate paths using three algorithms, including real-coded DE, GSA and IGSA. It is noteworthy from Fig. 18 that IGSA possess least ATTPT in comparison to the algorithms irrespective of the number of robots.

The performance analysis has been performed in terms of AUTTD over the number of steps in Fig. 19. It provides graphs between AUTTD vs. no. of stages required during the planning of path using three algorithms with number of obstacles = 7 and no. of robots = 5. It is apparent from Fig. 19 that AUTTD returns the smallest value for IGSA irrespective of the number of planning steps.

The performance of the result has been analyzed by plotting the average total trajectory path distance (ATTPD) with the number of robots as variable in Fig. 20. This path is generated by three different evolutionary algorithms such as GSA, DE, IGSA. Fig. 20 shows the result of ATTPD computation, when the number of robots varies between 1–5. Here, we observed that IGSA performs better than the other two algorithms as ATTPD is smallest for IGSA in comparison to other two algorithms irrespective of the number of robots.

Now, the performance analysis was undertaken by comparing the running time over the maximum number of iterations using three algorithms. Fig. 21 provides the time required for robots to reach in their respective goal position by three different algorithms and it shows that IGSA takes less time for robots to reach in destination.

Table 2
Description of obstacles presents in Fig. 4.

Robot number	No. of step required to goal in IGSA	No. of step required to goal in GSA	No. of step required to goal in DE
1	17	19	23
2	21	25	29
3	15	23	27
4	26	29	30
5	12	14	17

Table 3
Comparison of number of steps taken, ATTPT and ATTPD of different algorithms for different no. of robots.

No. of robots	Algorithms (steps taken)			ATTPT (in inch.)			ATTPD (in inch.)		
	IGSA	DE	GSA	IGSA	DE	GSA	IGSA	DE	GSA
2	12	16	18	35.7	36.5	38.4	3.7	4.7	5.7
3	15	18	20	37.8	38.6	40.4	4.9	5.6	6.6
4	19	21	24	39.7	40.5	42.6	6.8	7.3	7.9
5	**21**	**24**	**26**	**41.3**	**44.6**	**45.7**	**7.6**	**8.4**	**9.3**

Finally, the performance of the simulation result is analyzed in the terms of the number turn, by which we can able to minimize the energy consumption. The number of turn required for three different algorithms for number of robots = 6 is demonstrate in Fig. 22. It shows that IGSA takes less number of turn than other two algorithms and energy consumption to reach in the designation is less than the other two algorithms. The simulation is only presented for five numbers of robots but number of turn is less for irrespective of the robot in the planning scheme of the algorithm.

The experiment is conducted in the environment shown in Fig. 4 by the three algorithms for same fitness function in Eq. (29) with same parameter for 30 iteration, the best fitness value for three algorithms is presented in Fig. 23. The fitness value of the robots presented in Fig. 23 indicates that there is no conflict in the next position calculation by the robots, it shows that the best fitness value obtain for IGSA after 26 iteration is 3.638, but that achieved by GSA and DE after 29 and 30 is 4.105 and 4.711 respectively. This presents that IGSA is better than GSA and DE in the terms of avoiding problem at local optima and faster convergence rate.

Number of optimal steps reqiured for different robots, number from 1 to 5 of the simulation result for different algorithm is presented in Table 3. Table 3 shows that the number of optimal steps required for IGSA is less than the other algorithm such as GSA and DE. The total number of optimal steps required for IGSA, GSA and DE is 26, 29 and 30 respectively.

The result of the experiments performed is summarized in Table 2 in the terms of three performance metrics, namely, (1) total no. of steps required to reach in the goal, (2) ATTPT and (3) ATTPD have been used here to determine the relative merits of IGSA over the other algorithms for different robots. Table 1 confirms that the remaining two algorithms perform well with respect to all three metrics for different robots.

7. Experiment on Khepera II robot

Khepera II (Fig. 10) is a miniature robot (diameter of 8 cm) equipped with 8 built-in infrared range and light sensors, and 2 relatively accurate encoders for the two motors. The range sensors are positioned at fixed angles and have limited range detection capabilities. The sensors are numbered clockwise from the leftmost sensor 0 to sensor 7 and its internal structure (Fig. 12). Sensor values are numerical ranging from 0 (for distance > 5 cm) to 1023 (approximately 2 cm). The on board microprocessor has a flash memory size of 256KB, and the CPU of 8 MHz. Khepera can be used on a desk, connected to a workstation through a wired serial link. This configuration allows an optional experimental configuration with everything at hand: the robot, the environment and the host computer. The Khepera II network and its accessories are presented in Fig. 11 for the conduct of experiments.

The initial world map for conducting the experiment in the Khepera II is presented in Fig. 13 to 8 obstacles of different shape and predefine initial state and goal is marked on the map, where different meta heuristic algorithm

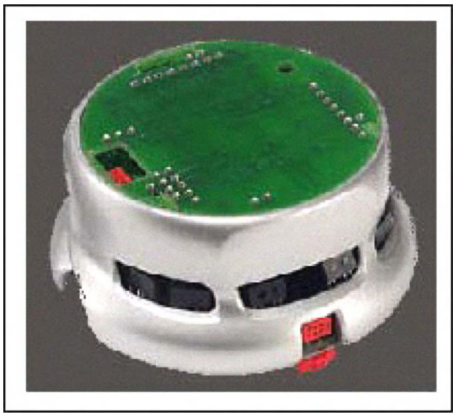

Fig. 10. The Khepera II robot.

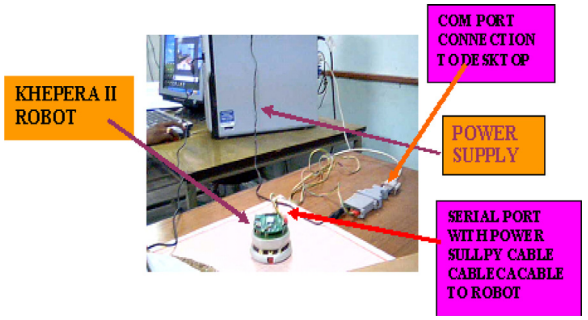

Fig. 11. Khepera network and its accessories.

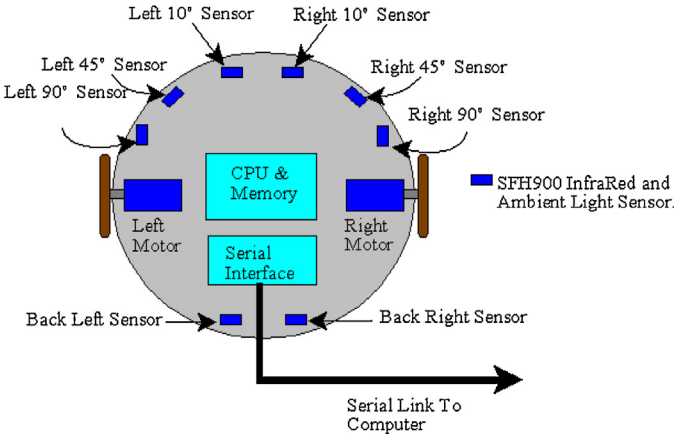

Fig. 12. Position of sensors and internal structure of Khepera II.

is applied. Fig. 14 shows the intermediate moment of the robot in the trajectory path toward the goal by respective robot using IGSA. IGSA is implemented in the Khepera-II robot with considering two robots and compared with a different evolutionary computing algorithm is demonstrated in Fig. 15. It shows better convergence in comparing to the other meta-heuristic algorithm presented in Fig. 15. Finally, different meta-heuristic algorithms have been applied in Khepera environment and results of the trajectory path have been presented in Fig. 15.

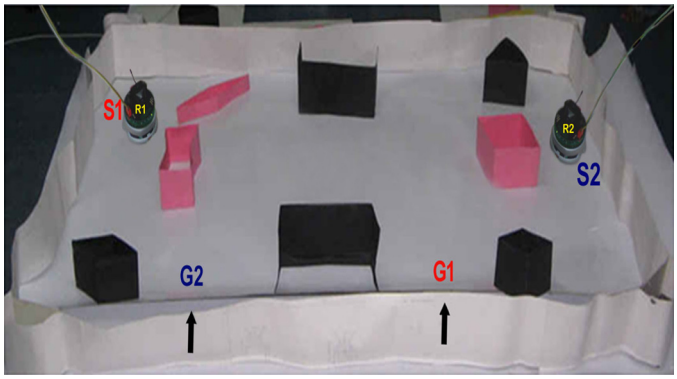

Fig. 13. Khepera environment setup for multi-robot path planning.

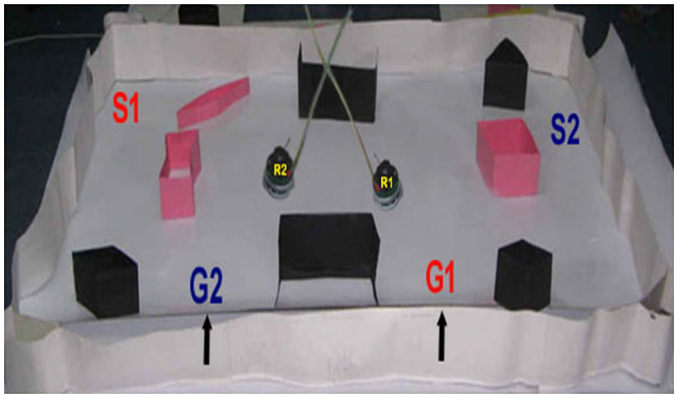

Fig. 14. Snapshot of intermediate stage of the multi-robot path planning using IGSA in Khepera environment.

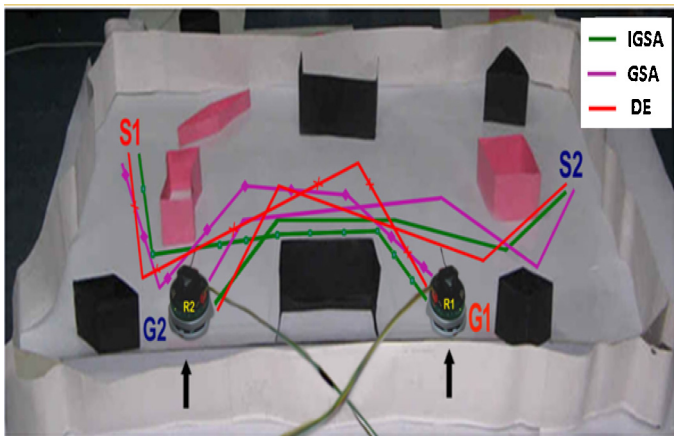

Fig. 15. Optimal path representation of different algorithm for multi-robot path planning in Khepera environment is represented by different color code.

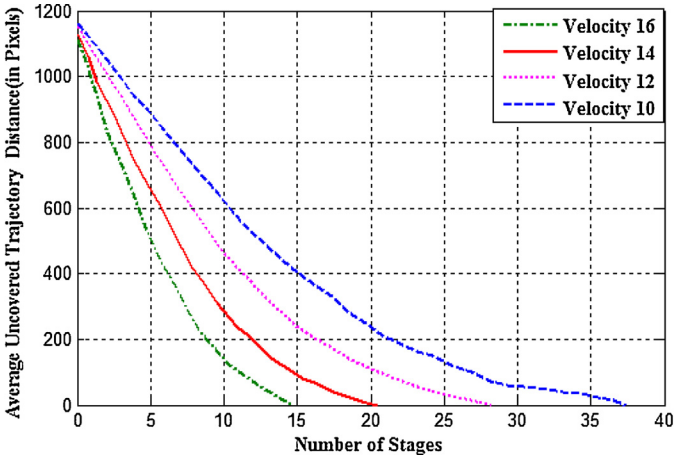

Fig. 16. Average uncovered trajectory distance vs. number of stages with variable velocity for fixed number of obstacles = 7.

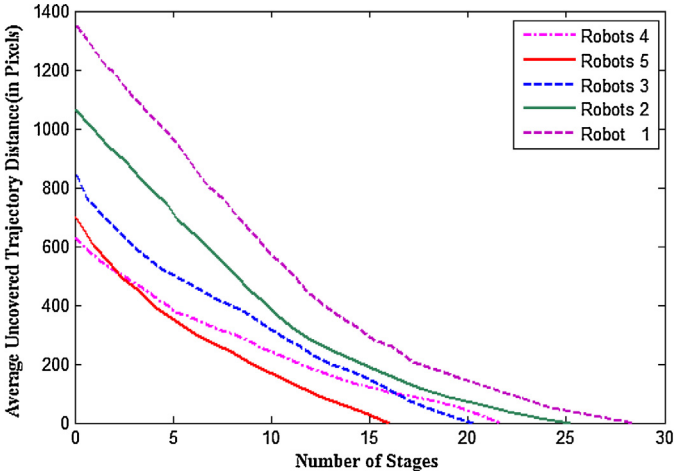

Fig. 17. Average uncovered trajectory distance vs. number of stages with variable number of robots for fixed number of obstacles = 7 (constant).

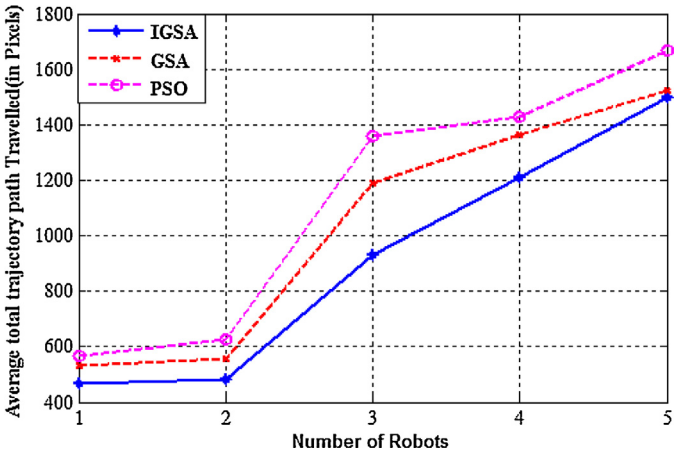

Fig. 18. Average total trajectory path traversed vs. number of robots with variable number of obstacles for fixed velocity.

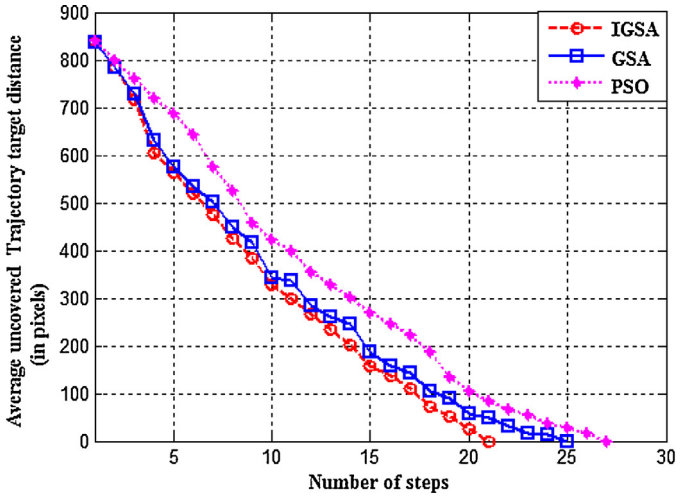

Fig. 19. Average uncovered trajectory target distance vs. number of steps in different algorithms.

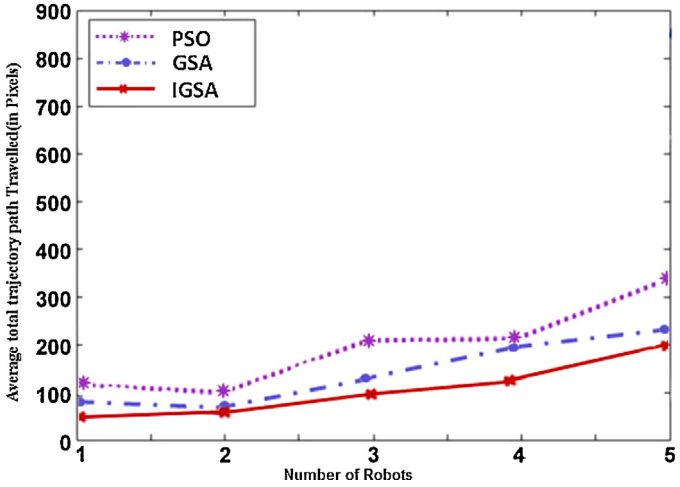

Fig. 20. Average total trajectory path deviation vs. no. of robots algorithm with fixed no. of obstacles = 7.

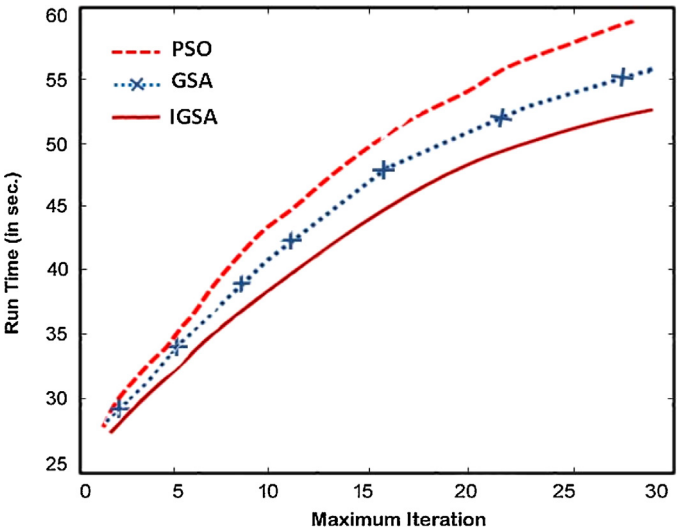

Fig. 21. Run time vs. maximum iteration for different algorithms.

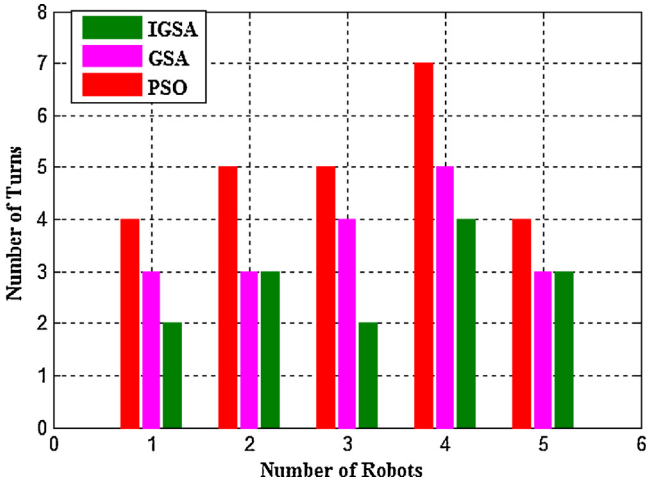

Fig. 22. Number of turn vs. number of robots in three different algorithms.

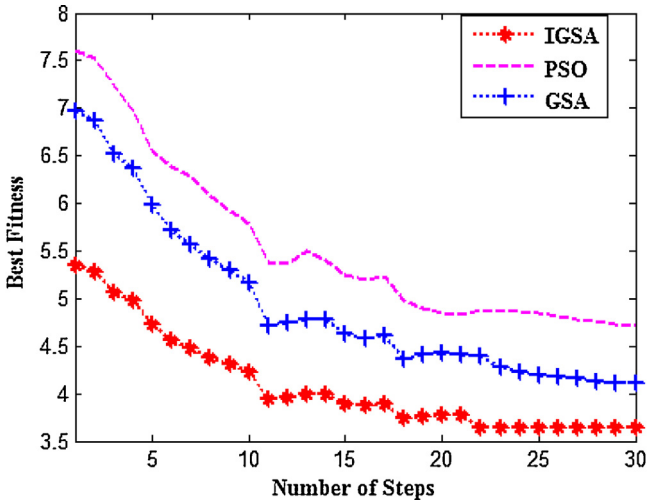

Fig. 23. Fitness value of IGSA, GSA and DE for fitness function in Eq. (29).

8. Conclusion and future works

An improved gravitational search algorithm was proposed for trajectory path planning of multi-robots in order to find collision free smoothness optimal path from predefine start position to end position for each robot in the environment. The obtained results from the experimental work perform better compared with the proposed algorithm. Comparing the performances among different techniques have been carried out. From the simulation and Khepera-II environment, it is observed that the IGSA technique is best over other technique for navigation of multi-mobile robot. However, in this paper, both the environment and obstacles are static relative to the robots; where as other robots are dynamic for priority robots. In future, work will be carried out using dynamic obstacles other than robots such as running vehicle, animals and on board camera during multi-robot path planning.

References

Alba, E., Dorronsoro, B., 2005. The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. IEEE Trans. Evol. Comput. 9 (2), 126–142 (IEEE).

Banerjee, G., Chowdhury, S., Losert, W., Gupta, S.K., 2011. Realtime path planning for coordinated transport of multiple particles using optical tweezers. IEEE Trans. Autom. Sci. Eng. 9 (October), 669–678 (IEEE).

Bhattacharjee, P., Rakshit, P., Goswami, I., Konar, A., Nagar, A.K., 2011. Multi-robot path-planning using artificial bee colony optimization algorithm. Proceedings of IEEE Congress on Nature and Biologically Inspired Computing, IEEE, 219–224.

Chakraborty, J., Konar, A., Jain, L.C., Chakraborty, U.K., 2009. Cooperative multi-robot path planning using differential evolution. J. Intell. Fuzzy Syst. 20 (1–2), 13–27 (IOS Press, Amsterdam).

Chatterjee, Mahanti, G.K., Mahapatra, P.R.S., 2011. Generation of phase-only pencil-beam pair from concentric ring array antenna using gravitational search algorithm. Proceedings of International Conference on Communications and Signal Processing, IEEE, 384–388.

Das, P.K., Konar, A., Laishram, R., 2010. Path planning of mobile robot in unknown environment. Int. J. Comput. Commun. Technol. 1 (2–4), 26–31 (Inderscience).

Das, S., Mukhopadhyay, A., Roy, A., Abraham, A., Panigrahi, B.K., 2011. Exploratory power of the harmony search algorithm analysis and improvements for global numerical optimization. IEEE Trans. Syst. Man Cybern. B 41 (1), 89–106 (IEEE).

Das, P.K., Pradhan, S.K., Patro, S.N., Balabantaray, B.K., 2012. Artificial immune system based path planning of mobile robot. Soft Comput. Tech. Vis. Sci. 395, 195–207 (Springer).

Eldos, T., Qasim, R.A., 2013. On the performance of the gravitational search algorithm. Int. J. Adv. Comput. Sci. Appl. 4 (8), 74–78 (The Science and Information Organization).

Geem, Z.W., Kim, J.H., Loganathan, G.V., 2001. A new heuristic optimization algorithm: harmony search. Simulation 76 (2), 60–68 (Sage).

Gong, C., Lincheng, S., 2001. Genetic path planning algorithm under complex environment. Robot 23 (1), 40–43 (Hindawi).

Guo, Y., Parker, L.E., 2002. A distributed and optimal motion planning approach for multiple mobile robots. Proceeding of IEEE International Conference on Robotics and Automation (ICRA'02), IEEE vol. 3, 2612–2619.

Kcymeulcn, D., Decuyper, J., 1994. The fluid dynamics applied to mobile robot motion: the stream field method. Proceedings of IEEE International Conference on Robotics and Automation, IEEE, 378–385.

Konar, 1999. Artificial Intelligence and Soft Computing: Behavioral and Cognitive Modeling of the Human Brain, 1st edition. CRC Press.

Li, C., Bodkin, B., Lancaster, J., 2009. Programming Khepera II robot for autonomous navigation and exploration using the hybrid architecture. Proceedings of 47th Annual South East Regional Conference, ACM, p. 31.

Liu, S. Hsi, Mernik, M., Hrnclc, D., Crepinsek, M., 2013. A parameter control method of evolutionary algorithms using exploration and exploitation measures with a practical application for fitting Sovova's mass transfer model. Appl. Soft Comput. 13 (9), 3792–3805 (Elsevier).

Luh, G.C., Cheng, W.-C., 2002. Behavior-based intelligent mobile robot using immunized reinforcement adaptive learning mechanism. Adv. Eng. Inform. 16 (2), 85–98 (Elsevier).

Masehian, E., Sedighizadeh, D., 2010. Multi-objective PSO- and NPSO based algorithms for robot path planning. Adv. Electr. Comput. Eng. 10 (4), 69–76 (Stefan cel Mare University of Suceava, Faculty of Electrical Engineering and Computer Science).

Precup, R.E., David, R.C., Petriu, E.M., Preitl, S., Radac, M.B., 2012. Novel adaptive gravitational search algorithm for fuzzy controlled servo systems. IEEE Trans. Ind. Inform. 8, 791–800 (IEEE).

Purcaru, C., Precup, R.-E., Iercan, D., Fedorovici, L.-O., David, R.-C., Dragan, F., 2013. Optimal robot path planning using gravitational search algorithm. Int. J. Artif. Intell. 10, 1–20 (CESER).

Rashedi, E., Nezamabadi-pour, H., Saryazdi, S., 2009a. GSA: a gravitational search algorithm. Inf. Sci. 179 (13), 2232–2248 (Elsevier).

Rashedi, E., Nezamabadi-pour, H., Saryazdi, S., 2009b. GSA: a gravitational search algorithm. Inf. Sci. 179, 2232–2248 (Elsevier).

Rashedi, E., Nezamabadi-pour, H., Saryazdi, S., 2010. BGSA: binary gravitational search algorithm. Nat. Comput. 9, 727–745 (Springer).

Regele, R., Levi, P., 2006. Cooperative multi-robot path planning by heuristic priority adjustment. Proceeding of IEEE/RSJ International Conference on Robotics System, IEEE, 5954–5959.

Sabri, N.M., Puteh, M., Mahmood, M.R., 2013. A review of gravitational search algorithm. Int. J. Adv. Soft Comput. 5 (3), 1–39 (SCRG).

Sarafrazi, S., Nezamabadi-pour, H., Saryazdi, S., 2011. Disruption: a new operator in gravitational search algorithm. Sci. Iran. 18 (3), 539–548 (Elsevier).

Tuncer, A., Yildirim, M., 2012. Dynamic path planning of mobile robots with improved genetic algorithm. Comput. Electr. Eng. 38 (6), 1564–1572 (Elsevier).

Verma, O.P., Sharma, R., Kumar, M., Agrawal, N., 2013. An optimal edge detection using gravitational search algorithm. Lect. Notes Softw. Eng. 1 (2), 148–152 (Springer).

Yang, X.-S., 2009. Harmony search as a metaheuristic algorithm. In: in Geem, Z.W. (Ed.), Music-inspired Harmony Search Algorithm: Theory and Applications, Studies in Computational Intelligence, vol. 191. Springer, Berlin, pp. 1–14.

Yu, J., Kromov, V., 2001. A rapid path planning algorithm of neural network. Robot 23 (3), 201–205 (Hindawi).

Zhang, Y., Gong, D.-W., Zhang, J.-H., 2013. Robot path planning in uncertain environment using multi-objective particle swarm optimization. Neurocomputing 103, 172–185 (Elsevier).